Using Decentralized Conflict-Abduction-Negation in Policy-Making

Etienne Houzé[1,2], Ada Diaconescu[2] and Jean-Louis Dessalles[2]

[1]*EDF R&D, 7 boulevard Gaspard Monge, 91120, Palaiseau, France*
[2]*Télécom Paris, 19 place Marguerite Perey, 91120, Palaiseau, France*

Abstract

In many real-world events that would require additional regulation, the causal chain leading to the event can be hard to determine. This is partly due to the distribution of knowledge across multiple agents, the a-priori unknown number an competence of such agents and their heterogeneous expertise. In this case, coordination is key to the understanding of the phenomenon. In this paper, we informally describe a novel approach to analyze complex sequences. This method originates from the study of smart homes, where collaboration between heterogeneous components is required, too. Our proposal is named D-CAS, which stands for Decentralized Conflict-Abduction-Negation. It is a high-level process that coordinates components' expertise to generate an explanatory reasoning in smart homes. We transfer our smart home solution to socio-technical systems in general. We illustrate the general concept via two fictional example cases: i) an autonomous car crash and ii) a crime perpetrated after social media fake news. In both cases, we examine how D-CAN could manage the communications and be used as a general framework to formalize interactions between experts and organize the discussion, helping to unravel a multi-domain causal chain. Using D-CAN helps identifying causes and responsible and can thus be helpful in a broader perspective of policy-making, e.g. to audit the potential flaws of current legislation.

Keywords

Reasoning, Multi-agent system, Explanation

1. Introduction

1.1. Problem presentation

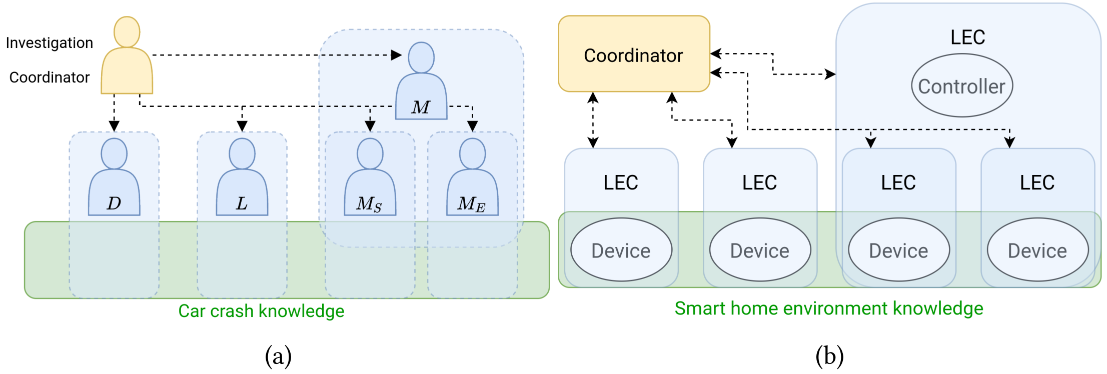
Policy makers often have to update regulation after tragic events occur. In such events, the causality chain and the responsibility of the different parties involved can be hard to identify, as knowledge of complex situations is scattered across experts with distinct domains. This can compromise the identification of efficient and fair regulation. To help the process, we propose to design a general solution to facilitate communication between experts and therefore the understanding of complex causal chains. We illustrate this via two fictional, yet realistic events. As they serve only as examples, we do not blame nor judge any involved party and do not draw any conclusion regarding the responsibility.

AMPM'21: First Workshop in Agent-based Modeling & Policy-Making, December 8, 2021, Vilnius, Lithuania
✉ etienne.houze@telecom-paris.fr (E. Houzé); ada.diaconescu@telecom-paris.fr (A. Diaconescu); jean-louis.dessalles@telecom-paris.fr (J. Dessalles)
🌐 https://explainableai.fr (E. Houzé); https://adadiaconescu.there-you-are.com (A. Diaconescu); https://perso.telecom-paristech.fr/jld/ (J. Dessalles)

In a first situation, suppose that an autonomous car, from manufacturer M has just crashed while its driver D was not using the wheel. This crash occurred at an intersection in a city that is supervised by some local authority L. The car manufacturer has two main branches in its production chain, the software team M_S and the car engineering team M_E. Considering the situation of the accident, precise knowledge about all elements of this crash can be considered to be split between these actors. The distinction is clear, meaning that no domain knowledge is sub-domain of another domain. In this situation, which is represented by Figure 1a, how can one understand the causal chain leading to the event of the crash? Should a new regulation introduce tighter control for the manufacturer, better road infrastructure or limitation to the driver's use of automated driving?

A similar example can be observed when considering a fictional crime. The police discovers that the main suspect had previously read and shared fake news on a social media platform, in a private group. Here, expertise from psychiatrists, the social media group, other members of the private group or the suspect him/herself can help understanding the process that led to the crime. Unfolding the chain of events is key to prove whether the suspect is guilty or innocent and to understand how policy changes can be performed to avoid similar situations in the future.

Both these cases highlight the important role played by causal understanding in policy-making. However this knowledge can be hard to acquire in situation where different actors hold part of the information. Furthermore, the number and domains of the different experts vary on a case-by-case basis. Here, coordinated communication is key to the success of the causal unraveling. Our contribution is to use an existing algorithm used for smart home explanations, named D-CAN, where different heterogeneous devices contribute to a system-wide explanation. D-CAN is based on an previously established cognitive process, CAN.



(a)

(b)

Figure 1: (a) The case of the car crash, where knowledge is distributed across multiple experts. (b) The architecture of a smart home enabling our D-CAN algorithm: each device and controller is associated with a Local Explanatory Component (LEC). In both cases, entities in blue are considered domain expert agents.

1.2. Related works

The investigation of complex causal chains is not new: different methods already exist to account for the investigation of past cases. For instance, the Why-Because Analysis (WBA) which is often used in reports following aircraft crashes [8]. However, to the best of our knowledge, they are more often used to explore and validate an investigation after its completion rather than being used as a driving mechanism for the reasoning.

The notion of causality is the basis of WBA explanation. Recent advances in the study of this phenomenon show the importance of *"contrastive explanation"* [6, 5] where the object of the explanation is the difference between an expected state of affairs and the actual observation. From this basis, we can draw a parallel between explanation and spontaneous argumentative reasoning [1] each sentence is relevant because it tries to solve an existing conflict.

In a broader perspective, STAMP (System Theoretic Accident Model)[3] proposes an accident model that is based on systems and control theories. It incorporates an event-based causal modeling at the level of the process itself, but incorporates it in a larger model, encompassing system development and system operations, that are modeled as multi-layered control loops. This approach can model real-life accidents, such as airplane failures [4], but it requires expert knowledge from the investigator, spanning the generation, the process itself and its monitoring.

A similar issue may arise when considering the standard theory of Argumentation which is mainly due to Dung's formalism [7]. In this framework, a binary relation of attack is defined over a set of arguments, which allows to define which arguments are acceptable to defend which proposition. While this theory models formal argumentative debates such as law or medicine, it fails to encompass the runtime changes of beliefs or arguments that exist in mundane argumentation. In addition, knowing all arguments and attack relations requires a high level of expertise, which may be impossible to maintain for topics covering several areas.

2. Decentralized Conflict-Abduction-Negation

2.1. Principles of CAN

The *Conflict-Abduction-Negation* (CAN) process was first designed as a generic cognitive model to account for argumentative reasoning[1].

At the core of CAN lies the notion of *conflict*[1]: a conflict occurs when an observation is in contradiction with a prior belief or desire of the agent. To formalize the notion of conflict, we represent the agent's perception of the world as a set of Boolean predicates P. In addition, an agent can associate a necessity $v(P)$ to each predicate P. This necessity $v(P)$ is a number conveying information about the strength and nature of the agent's prior opinion regarding P's value. A positive necessity means that the agent wishes or expects P to be true, while $v(P) < 0$ means that the agent wishes or expects it to be false. The internal state of the agent's perception of the world and its opinions can be modeled by a set of predicates and necessities. Note that a necessity set to 0 means that the agent has no opinion regarding a predicate.

A conflict occurs when the predicate's value contradicts the sign of the associated necessity. For instance, P can be true while its associated necessities $v(P)$ is negative. The couple $(P, N = v(P))$ models the conflict, whose *intensity* is given by the absolute value $|N|$. Similarly, a conflict can occur if a predicate P is observed to be false while its associated necessity is positive. It is also important to note that if (P, N) is a conflict, then its negation $(\neg P, -N)$ also constitutes a conflict of the same intensity.

CAN relies on the identification of conflicts (P, N) which can then be propagated to either their causes, consequences or negations. i) *Abduction* denotes the process of inferring a causal hypothesis to an observed phenomenon. In CAN, this means that, if C is found to be the cause of P, then the conflicts (P, N) is propagated onto (C, N); ii) *Negation* allows to consider potential actions and alternative scenarios by considering the opposite of a conflict.

CAN can be considered an alternative to the classic Argumentation Theory [1, 7]. It provides a minimalist model, where conflicts and their relevance is evaluated at runtime rather than in a formal *attack/defend* classification. It is, however, compatible: the output of CAN can be represented as a set of formal arguments and attacks, following the inference knowledge shown by the procedure.

2.2. Decentralization

D-CAN extends this process by observing that knowledge, i.e. predicates values and their associated necessities, can be only locally defined and therefore a inter-agent process is possible without relying on an omniscient knowledge base. A coordinator is necessary to maintain coherence between calls and provide a unique interface with the user, but it can be kept minimal and generic, in the sense that it simply routes requests to the relevant expert component. This means that in the smart home for instance, when requested to give an explanation for a low temperature, the coordinator component will route the request to the temperature controller component, which will inquire the conflict by observing its measures and trying to identify a possible cause thanks to its local knowledge. The request is then sent back to the coordinator, which will once again route it to the next component.

The detailed process is shown in Algorithm 1. The coordinator identifies the expert responsible for the conflict proposition P, then asks it to inspect the problem with its local knowledge (line 6) and waits for its answer. This latter can be either a causes inferred via inference, a consideration of the negation of the current conflict, or the abandon of the conflict. The coordinator updates its knowledge accordingly: this eventually propagates the conflict onto the inferred cause, the negation or another conflict occurring elsewhere in the system. The algorithm terminates once all conflicts have been examined and handled, solved or discarded.

Figure 1b illustrates a possible Smart Home architecture to enable D-CAN: a central coordinator is connected to Local Explanatory Components located on several low-power computers, each acting as a local expert in the knowledge domain covered by its associated device. This allows the process to be decentralized, in the sense that the knowledge of the system is distributed among various experts. Note that contrary to our previous work[2], we use D-CAN instead of D-CAS (where Simulation replaces Negation). The motivation for this change is that cyber-physical systems can not simply deny an observation but instead can run a simulation. Since we are considering a generalization to other domains here, this modification is irrelevant, hence the use of D-CAN.

Algorithm 1: The D-CAN algorithm

```
Input: A request $(P, N)$
Result: A conflict-solving process
responsible $\leftarrow$ locateResponsible(P);
while responsible $\neq$ $self$ do
    if responsible = null then
        Backtrack() ;
    end
    answer = responsible.investigate((P, N)) ;
    switch Answer do
        case ABDUCTION do
            (P, N) $\leftarrow$ Answer.Hypothesis ;
            responsible $\leftarrow$ locate(P) ;
        end
        case GIVE UP do
            Backtrack();
        end
        case NEGATION do
            assessTrue(Answer.Action) ;
            Conflict $\leftarrow$ findConsequences() ;
        end
    end
end
```

3. Application to the examples

To illustrate how D-CAN can fare in real-life situation, we consider its application to the two examples presented in the introduction of this paper.

3.1. Playing the examples

For the car crash example, a possibility of a mental D-CAN unraveling is presented in Table 1: the successive calls between agents are represented, as well as the content of these calls, i.e. a conflict-like object. In this example, the trace shows the following discussion: the driver first thinks that the car failed to stop correctly, which might indicate a problem with the manufacturer M. The latter conducts internal examination but rules out the possibility of a mechanical or a software failure. The driver is then asked for more details about the situation, to find another cause for the crash. She thus indicates that the area was not correctly lit, which triggers an inquiry with the local authorities L. Note that the intensity of conflicts proposed by abductive reasoning is lower than that of the incoming conflict. This gap translates the uncertainty regarding the outcome of abduction. In addition, it prevents from ending up considering causal chains that are too long: after a number of abductive hypotheses, a low intensity prevent the

Sender	Target	Content	Comments
External	Coordinator	(crash, -50)	Initial request
Coordinator	D	(crash, -50)	Routing
D	Coordinator	(car_no_stop, -45)	Abductive reasoning
Coordinator	M	(car_no_stop, -45)	Routing
M	M_S	(software_issue, -40)	Abductive reasoning
M_S	M	(-software_issue, -40)	No problem found
M	M_E	(mechanical_issue, -40)	Other Abductive hypothesis
M_E	M	(-mechanical_issue, -40)	No problem found
M	Coordinator	(-car_no_stop, -45)	No problem found
Coordinator	D	(crash, -50)	Asking again
D	Coordinator	(-light, 40)	Second hypothesis
Coordinator	L	(-light, 40)	Inquiring Local authorities
		...	

Table 1
Successive calls in the example of the car crash

conflict to get propagated. In case he/she wants more information, the coordinator can ask again with more intensity, which will propagate the conflict further.

Sender	Target	Content	Comments
External	Coordinator	(crime, -50)	Initial request
Coordinator	Suspect	(crime, -50)	Routing
Suspect	Coordinator	(fake_news, -45)	Abductive Reasoning
Coordinator	Social Media Company	(fake_news, -45)	Routing
Social Media Company	Coordinator	(- fake_news, -45)	Giving up
End of the first try. Trying again, with more intensity...			
External	Coordinator	(crime, -80)	Initial request
Coordinator	Suspect	(crime, -80)	Routing
Suspect	Coordinator	(fake_news, -75)	Abductive Reasoning
Coordinator	Social Media Company	(fake_news, -75)	Routing
Social Media Company	Coordinator	(hate_group, -70)	Abduction
Coordinator	Police	(hate_group, -70)	Routing
		...	

Table 2
Successive calls in the example of the fake-news crime

The second example, the fake-news-related crime, illustrates another possibility of D-CAN. As necessities encode the intensity of a conflict and the underlying request, it is possible that this intensity is not high enough to trigger some possibilities: think, for instance, of a test that would require a considerable amount of work, or an information one of the agents is not keen on disclosing. If the problem to solve appears minor, it might be better not to consider doing this test or revealing sensible information, as it would not be worth the trouble. In this case, the reasoning explores other branches of the reasoning first. This is what happens in the rationale exposed in Table 2: as it is costly for the social network company to investigate and disclose personal information, at first it does not answer the request for more information about the fake news. However, as no other branch is found in the rationale, the process can be re-launched, this time with a higher intensity, which will be sufficient to provoke a reaction from the social media company. Then, the reasoning can go on and, for instance, the police may be called to acquire more information about the original posters of the hate group on the social media platform.

3.2. A tree representation of reasoning

The main advantage of using D-CAN in deliberative discussions is normalization: all inquiries and relations are represented using the same grammar of predicates and necessities, as do agents' beliefs and reasoning. Therefore, it is possible to transcribe the reasoning and represent every step by a tree graph, whose nodes represents call. Figure 2 illustrates this by showing the tree representing the rationale of the car crash example detailed in Table 1. The tree visualization reads from left to right, and top to bottom. It shows each request to a different expert agent, the different examined conflicts and the outcomes of the calls. This visualization tool offers an option to display an interpretable description and keep a track of the different calls and conflicts, retracing the entire reasoning.

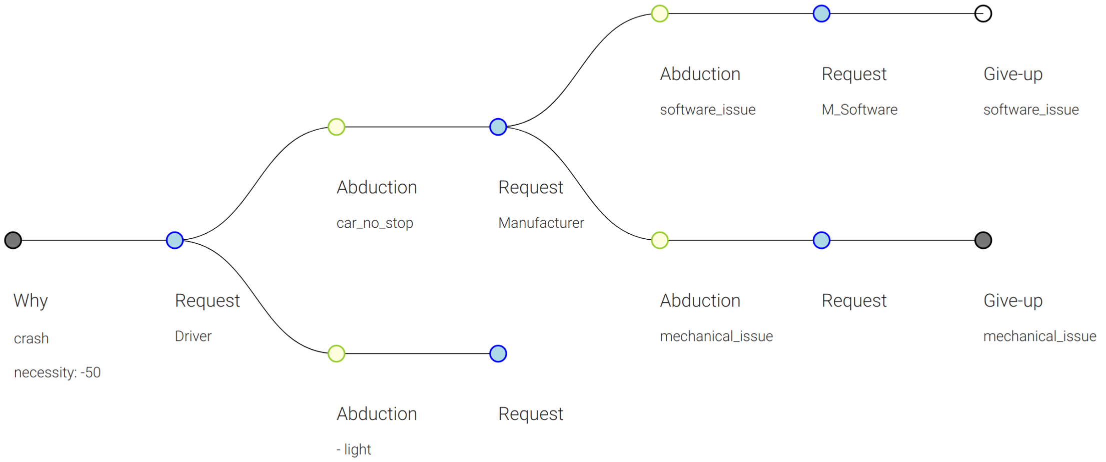


Figure 2: The tree representation of the D-CAN process for the example of the car crash.

4. Perspectives and conclusions

We propose a new approach on multi-agent exploration of causal chains for policy-making based on an existing process for smart-home systems, D-CAN. We use two basic but realistic examples to illustrate the ability of D-CAN to coordinate requests and formalize interactions, which allows to generate visualization of the reasoning.

This work can be considered as preliminary, as it is still in very early stage and many issues have not been addressed yet. For instance, we modeled agents in a simple manner to keep the minimal aspect of D-CAS. Thus, we represent requests as simple conflict-like objects and agent’s knowledge as a collection of instantiated predicates and necessities. While powerful and efficient in the case of smart home devices, this representation can prove insufficient for human agents, where other variables can be taken into consideration: deception, genuine mistakes and misunderstanding are part of any human organization. While D-CAN helps tackling the third of these issues, understanding how to support handling the former two remains for future development

References

[1] J.-L. Dessalles, “A Cognitive Approach to Relevant Argument Generation,” in Principles and Practice of Multi-Agent Systems, LNAI 9935, 2016, pp. 3–15.

[2] É. Houzé, J.-L. Dessalles, A. Diaconescu, D. Menga, and M. Schumann, “A Decentralized Explanatory System for Intelligent Cyber-Physical Systems,” in Proceedings of SAI Intelligent Systems Conference, 2021, pp. 719–738.

[3] N. Leveson, “A new accident model for engineering safer systems,” Safety science, vol. 42, no. 4, pp. 237–270, 2004.

[4] C. K. Allison, K. M. Revell, R. Sears, and N. A. Stanton, “Systems Theoretic Accident Model and Process (STAMP) safety modelling applied to an aircraft rapid decompression event,” Safety science, vol. 98, pp. 159–166, 2017.

[5] R. R. Hoffman and G. Klein, “Explaining explanation, part 1: theoretical foundations,” IEEE Intelligent Systems, vol. 32, no. 3, pp. 68–73, 2017.

[6] T. Miller, “Explanation in artificial intelligence: Insights from the social sciences,” Artificial Intelligence, 2018.

[7] P. M. Dung, “On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games,” Artificial intelligence, vol. 77, no. 2, pp. 321–357, 1995.

[8] P. Ladkin and K. Loer, “Analysing aviation accidents using WB-Analysis -– An application of multimodal reasoning,” 1998.

[9] R. R. Hoffman and G. Klein, “Explaining explanation, part 1: theoretical foundations,” IEEE Intelligent Systems, vol. 32, no. 3, pp. 68–73, 2017.